ARMY RESEARCH LABORATORY

Tactically Significant Route Planning

George W. Hartwig, Jr.
Frederick S. Brundick
CPT(P) Scott D. Kothenbeutel

ARL-TR-1139 July 1996

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1996	Final, Oct 93--Sep 95

4. TITLE AND SUBTITLE

Tactically Significant Route Planning

5. FUNDING NUMBERS

4T592502T4 3004
CC: 4T2000

6. AUTHOR(S)

George W. Hartwig, Jr., Frederick S. Brundick, CPT(P) Scott D. Kothenbeutel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-IS-TP
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER

ARL-TR-1139

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

A methodology introducing the Shape of Certainty (SOC) as a partial solution to the problem of situational awareness on the AirLand battlefield is presented. Applying the three tenets of Information Distribution Technology (IDT) involves sending information: (1) in its most general form, (2) only when necessary, and (3) in an efficient manner, in an attempt to distribute positional data throughout the battlefield as well as indicate how current positions match the intended order of battle. The process involves making use of preplanned dynamic axes of advance, distributing these axes, and limiting information exchanges to those occasions when deviations occur. This work is done in support of the Combined Arms Command and Control (CAC2) Advanced Technology Demonstration in cooperation with the Army Communications-Electronics Command (CECOM).

14. SUBJECT TERMS

maneuver planning, situational awareness, tactical communications

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

LIST OF FIGURES

INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

> "The incident was tragic. An immediate investigation revealed that both units were on the correct side of their respective boundaries. Extreme darkness created by cloud cover made visibility possible only through night vision devices. While thermals could be used for precise targeting out to a range of 3.2 km, the thermal image was so indistinct that beyond about 700 m gunners found it difficult to differentiate between friend and foe. A square plywood tool box mounted in the back of one of the larger engineer vehicles made it appear through the thermals to be a building."
>
> "Fear of further fratricide grew as word of incidents and near misses filtered throughout the ARCENT chain of command. The corps commanders became increasingly concerned as their units began to converge in the tightly constricted battle space of Kuwait. Among thousands of units from platoons through corps, extreme caution began to work its way into the execution of the battle plan. Attacks halted more frequently to allow units to sort themselves out before advancing again. Distant targets, most surely Iraqi, were allowed to escape without engagement. [Generals] Luck and Franks established a 5-km "sanitary" zone between their respective corps and agreed that no target in the zone, even if positively identified as Iraqi, would be engaged. Undoubtedly all these actions saved lives, but the price paid for more safety was a substantial increase in friction and concomitant dampening of audacity and dash throughout the remainder of the campaign." (Scales 1993)

With these words, MG Scales and the "Mail House Gang" defined the problems modern technology introduces to the battlefield and the impact on AirLand battle doctrine. Not only is fratricide a tragic reality, but the ensuing restraint of maneuver may put the entire plan in jeopardy.

1.1 Background. The Military Computer Science Branch of the U.S. Army Research Laboratory (ARL), in cooperation with the Army Communications-Electronics Command (CECOM) as part of the Combined Arms Command and Control Advanced Technology Demonstration (CAC2ATD), has embarked on a program introducing the *dynamic axes of advance* as a partial solution to the problem of situational awareness on the AirLand battlefield. This program will apply the Information Distribution Technology (IDT) tenets of sending information,

- in its most general form,
- only when truly necessary, and
- in an efficient manner,

in an attempt to distribute positional data throughout the battlefield as well as indicate how current positions match the intended order of battle. The key to this process is to make use of preplanned

dynamic axes of advance. Once these axes of advance are created and properly distributed, it is relatively straightforward to limit information exchanges to those occasions when deviations occur. What constitutes a deviation would be determined by commanding officers and/or standing operating procedures. It might be as simple as staying within a specified distance of where your plan says you will be.

1.2 Example. Let us assume that a unit has been given a mission that requires them to traverse the axis of advance as shown in Figure 1. Here we have an axis of advance with phase lines. If we associate times with the phase lines, we would be able to estimate, using simple interpolation and within limits, where the unit should be located at any time during the mission.* As long as the schedule is kept, there would be no need to transmit position updates, since everyone interested could estimate where the unit is located. Only when the unit deviates from the plan is the unit commander advised to act. This is shown in Figure 2 where the ARL concept is presented. In this case, a route has been determined that would approximate the center of the axis of advance. Since every unit needs room to respond to local conditions, an area of maneuver, or shape of certainty (SOC), is associated with the route. The "SOC" is so named since we intend to guarantee that the unit is within this area. The width of the SOC could correspond to current axis of advance. As Figure 3 shows, the ARL plan is equivalent to an axis of advance; however, since we have Global Positioning System (GPS) units and computers to help, the use of the SOC allows us to be more precise in specifying where a unit should be located at any particular time. The use of the SOC allows the unit commander the leeway necessary to accommodate the tactical situation.

2. THE SHAPE OF CERTAINTY (SOC)

Situational awareness may be enhanced by defining a sequence of objectives for each unit or collection of units. Each of these objectives, or waypoints, has a location and definite time associated with it. A unit proceeds from point to point, arriving at the specified time, keeping its speed constant within each segment. At a given time, it is possible to calculate *exactly* where a unit should be by determining which segment it is on, then interpolating based on time.

However, units will never perform exactly as planned, so a tolerance box or SOC must be placed around the unit's desired location. A unit that is anywhere within this area will be considered to be following its plan.

---

* The limits would depend on many tactical factors, including terrain, weather, commander's guidance, etc.

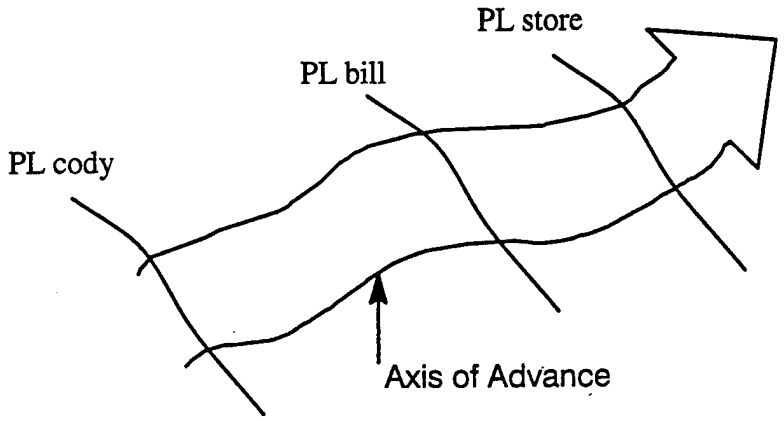

Figure 1. Axis of advance with phase lines.

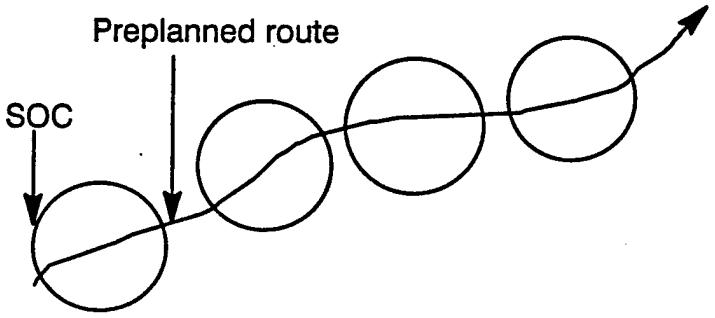

Figure 2. Preplanned routes.

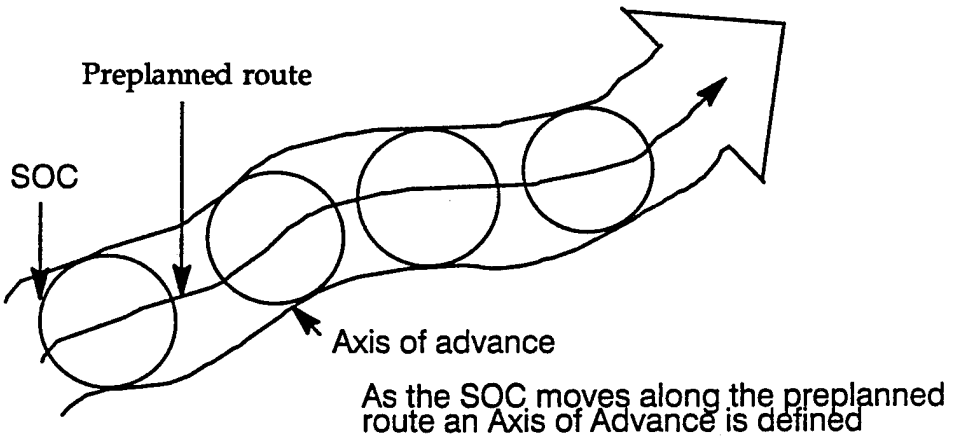

Figure 3. The SOC traces out an axis of advance.

2.1 Complications. To keep the computations relatively simple, rectangles were chosen for the SOC. Circles are even easier, but finer control is needed over the shape. Rectangles permit an increase in the allowable error in the direction of travel while limiting the error to the sides, while a circle grows equally in all directions. The unit's location may also be offset from the center of the SOC; this can be done with a circle but is not intuitive.

The rectangle for each segment is defined in terms of four parameters:

(1) **E** is based on the early time, or "no earlier than" time. The user will supply the early time, and the objective planning program will convert that into a distance.

(2) **L** is based on the late time, or "no later than" time. It is analogous to **E**, except it is a distance behind the unit, not in front.

(3) **P** is the distance to the left of the unit.

(4) **S** is the distance to the right of the unit. Normally **P** and **S** will be equal unless terrain or adjacent units cause the SOC to be shifted to one side.

This report will refer to the distances D_E, D_L, D_P, and D_S. A dimensioned SOC is shown in Figure 4. The unit is traveling to the right.

2.2 Waypoints. As a unit approaches a waypoint, its SOC must gracefully blend into the SOC for the next segment. A unit that has been inside of its SOC along one segment should not suddenly find itself outside the new SOC as the predicted location passes the waypoint.

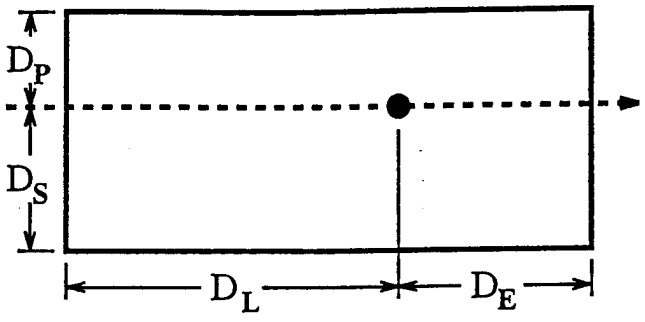

Figure 4. Dimensioned SOC.

The SOC at a waypoint may be perceived as two SOCs, one for each segment. The SOC on the first segment is a rectangle with a semicircle affixed to the front, while the second SOC has a semicircle extending from the back of the rectangle. If the widths of the two SOCs are similar, the area of overlap works quite well. Such a case is shown in Figure 5. The shaded area is the additional blending generated by the two semicircles. Without this area, a unit passing the waypoint on the left side of its route would be in a gap outside the SOC.

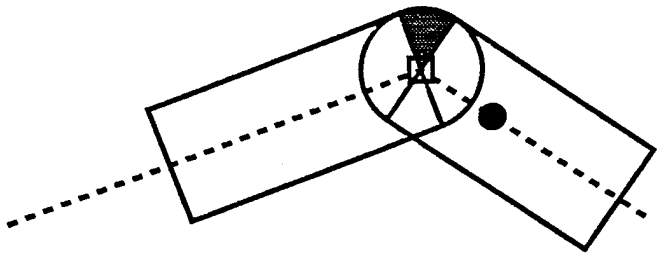

Figure 5. Waypoint inside SOC.

2.3 Calculations. While the unit's predicted location is away from a segment's endpoints, the length of the SOC is $D_E + D_L$. At some time, the front edge reaches the waypoint and the forward portion of the SOC begins to shrink; simultaneously, the SOC on the next segment begins to grow. Once the unit reaches the waypoint, the rearward portion starts to shrink on the first segment and grow on the second. Eventually, the rear edge reaches the waypoint and the entire SOC is on the second segment. This is shown in Figure 6.

Let the desired waypoint time be denoted by T_i. The front of the SOC will arrive at T_{NET_i}, and the back edge will arrive at T_{NLT_i}. In Figure 6(a), the front of the SOC has not arrived at the waypoint, so $T = T_{NET} - \epsilon$, where ϵ is some small amount of time, and in (b), the front has just passed the waypoint, so $T = T_{NET} + \epsilon$. The unit's location passes the waypoint in (c) and (d), where $T = T_i - \epsilon$ and $T = T_i + \epsilon$, respectively. Likewise, the end of the SOC nears the waypoint in (e) at $T = T_{NLT} - \epsilon$ and is entirely on the new segment in (f) at $T = T_{NLT} + \epsilon$.

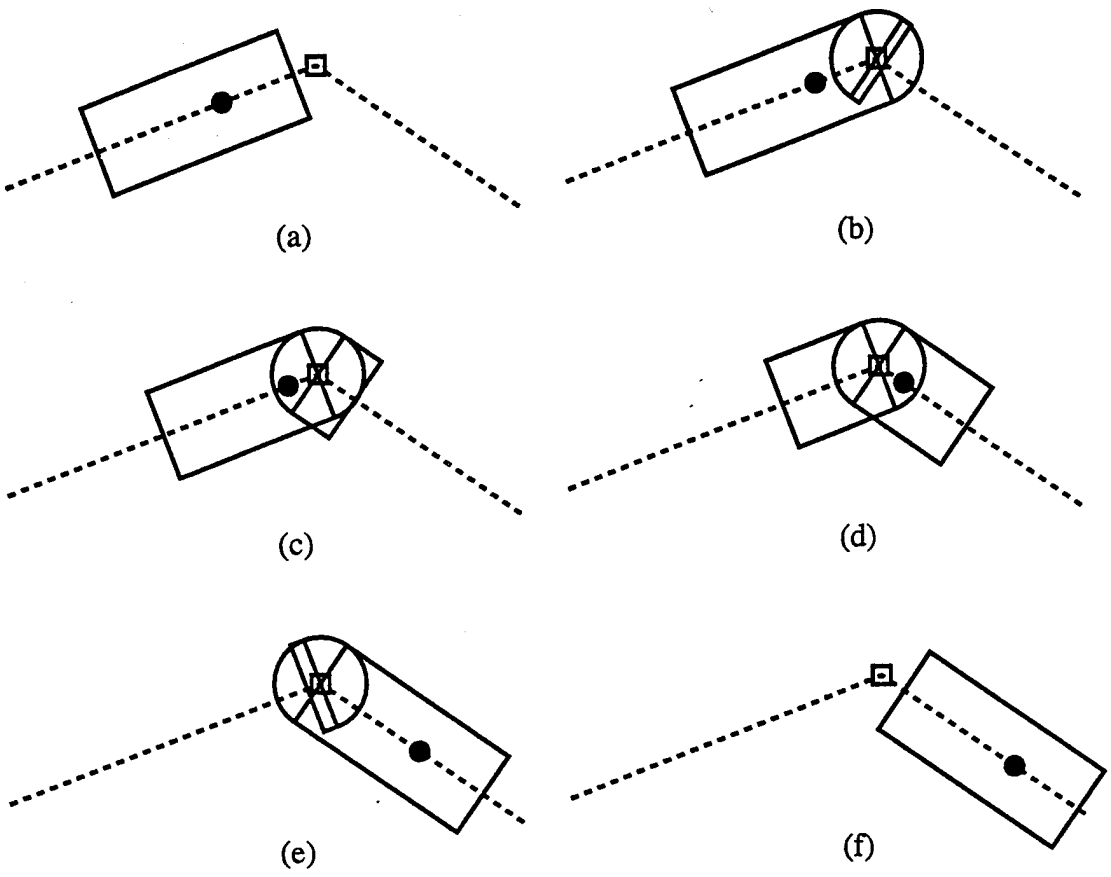

Figure 6. Unit passing a waypoint.

Before the times when the SOC first and last touches the waypoint can be computed, some decisions must be made about vehicle movement. The values used to generate Figure 6 were carefully chosen to keep everything constant (i.e., $V_i = V_{i+1}$, $D_{E_i} = D_{E_{i+1}}$, and $D_{L_i} = D_{L_{i+1}}$). The predicted unit location, front edge of the SOC, and rear edge all move at a constant velocity.

Suppose the unit moves at a constant velocity along each segment (it may change from segment to segment) and never stops, instantly changing its velocity from V_i to V_{i+1}. When it arrives at a waypoint, it immediately begins to move along the next segment. This implies that the earliest time the unit may leave a waypoint is the same as the earliest time it may arrive $\left(T_{NET_i}\right)$. However, in order for the unit to arrive at the next waypoint no earlier than its allowed time $\left(T_{NET_{i+1}}\right)$, the early time interval must be a constant, or $t_{E_i} = t_{E_{i+1}}$.

This constraint—the early time is a constant—is too restrictive. (By the same reasoning, the late time must also be a constant.) The time and distances are both fixed, so the solution is to allow the velocity to vary within certain bounds. A unit is shown approaching a waypoint in Figure 7. The SOC drawn above the route has just touched the waypoint at time T_{NET_i}, and the one below the route is at T_i. As time advances from T_{NET_i} to T_i, the early (front) portion of the first SOC shrinks from a length of D_{E_i} to 0, while, simultaneously, the early portion of the second SOC grows from 0 to $D_{E_{i+1}}$.

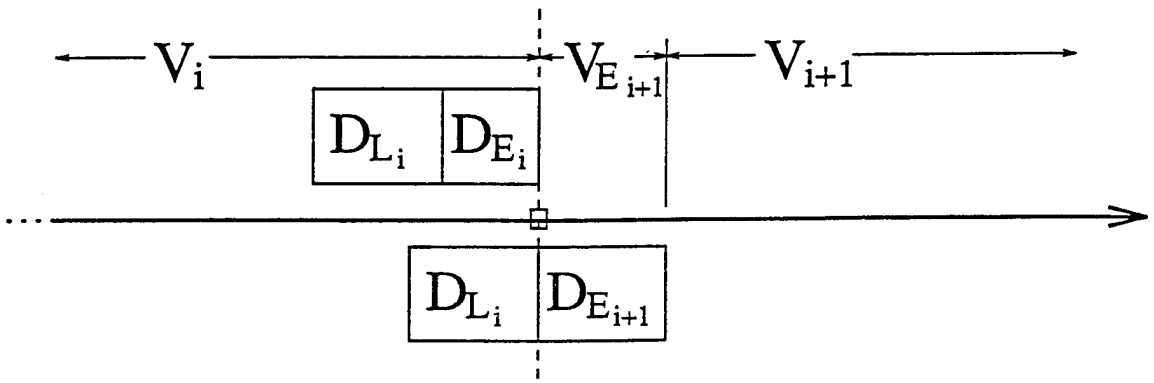

Figure 7. Early velocities.

The unit and front edge of the SOC are both advancing toward the waypoint at the velocity V_i. When the front edge reaches the waypoint, it changes to $V_{E_{i+1}}$ until it has covered the distance $D_{E_{i+1}}$. At this time, the unit has arrived at the waypoint and both the unit and front end begin moving at V_{i+1}. For the interval $T_{NET_i} \leq T \leq T_i$

$$t_{E_i}(arr) = \frac{D_{E_i}}{V_i} \tag{1}$$

and

$$t_{E_{i+1}}(dep) = \frac{D_{E_{i+1}}}{V_{E_{i+1}}} . \tag{2}$$

Combining the equations gives

$$t_{E_i}(arr) = t_{E_{i+1}}(dep) \tag{3}$$

or

$$\frac{D_{E_i}}{V_i} = \frac{D_{E_{i+1}}}{V_{E_{i+1}}}, \tag{4}$$

resulting with

$$V_{E_{i+1}} = V_i \times \frac{D_{E_{i+1}}}{D_{E_i}} . \tag{5}$$

A unit that is traveling at V_i near the predicted location will not be affected by these complications. However, a unit near the front of the SOC may need to change its velocity as it passes a waypoint as computed by equation (5). If $D_{E_{i+1}} < D_{E_i}$, the unit must slow down to prevent its arriving at the next waypoint too soon, while, if $D_{E_{i+1}} > D_{E_i}$, the unit may speed up. Waypoints frequently indicate a change in unit velocity which cannot occur instantaneously in the real world. The unit may be cresting a ridge, negotiating a turn, or even changing formation. Its velocity is certainly varying quite a bit, so the requirement that the front of the SOC also advance at a new speed is not a serious complication.

The situation for the movement of the rear of the SOC is similar to that of the front of the SOC as shown in Figure 8. The rear of the SOC moves a distance of D_{L_i}, while the predicted location moves a distance of $D_{L_{i+1}}$.

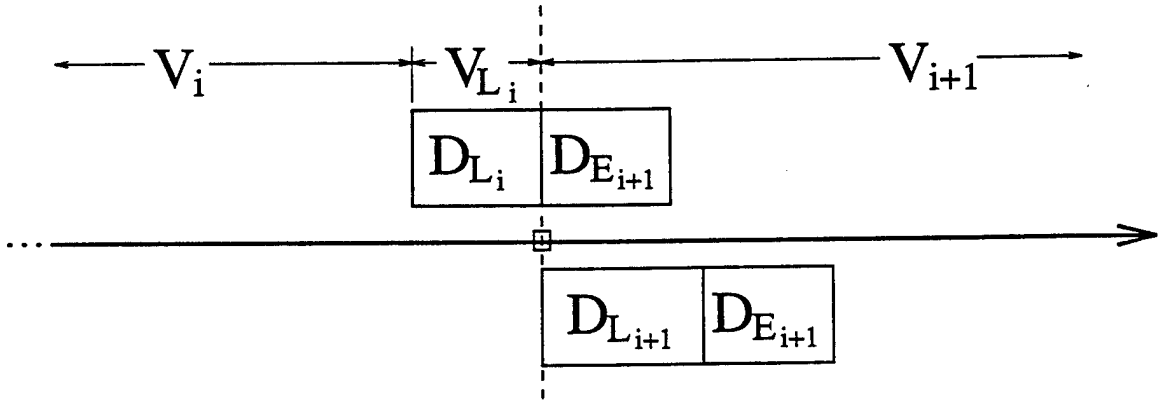

Figure 8. Late velocities.

This may be expressed as

$$V_{L_i} = V_{i+1} \times \frac{D_{L_i}}{D_{L_{i+1}}} . \quad (6)$$

2.4 Position Prediction. Once a unit has a planned sequence of objectives and tolerances, it may predict where it is supposed to be at a given time. The list of objectives is scanned to determine on which segment the unit should be; then a linear interpolation is performed to pinpoint the location. The current prototype program draws a circle about the point and checks to see if the unit's actual location (as provided by a GPS unit) is within the circle.

Actually, all four parameters should be used with a rectangular SOC. This requires that all the arrival and departure times be precomputed as shown in Figures 7 and 8. Not only must the unit's predicted location be determined, but the location of the front and rear edges of the SOCs must also be computed by interpolation. This is trivial if the unit is in mid-segment, and slightly more complex if the SOC is passing a waypoint. However, it can be very complicated if the entire SOC contains multiple waypoints.

Field experiments will be conducted to show that the SOC concept is practical. At the same time, other methods will be explored for handling the transition problems that occur at waypoints. What is needed is a quick, efficient way of determining if a unit is within its SOC. It may not be necessary to compute the entire area of the SOC except in graphical systems that show where units may be located. Approximate calculations may be adequate there, too.

3. FACT TYPE DEFINITION

Two of the details that must be resolved when dealing with positional data are: what coordinate system should be used and how much resolution, or precision, is required to specify those coordinates?

3.1 Coordinate System. Since the Earth is basically a sphere, spherical coordinates, or latitude and longitude, would seem to be the coordinate system of choice. For locating oneself on the globe, this is true. Unfortunately, for performing computations such as finding the distance between two points, latitude and longitudes are difficult to use and not at all intuitive.

For almost all Army applications, the plane approximation, using Universal Transverse Mercator (UTM) coordinates, is "good enough."* The other reason is that the computation required to actually find the "great circle" distance between two known points is complex. The needed equation is derived using spherical triangles

$$\cos D = \sin(L_1) * \sin(L_2) + \cos(L_1) * \cos(L_2) * \cos(DL_0),$$

where

L_1, λ_1 are the latitude and longitude of one point

and

L_2, λ_2 are the latitude and longitude of the second point,

then

$DL_0 = \lambda_1 - \lambda_2$
$M_1 = L_1, \lambda_1$
$M_2 = L_2, \lambda_2$.

D is in terms of the angle of the arc, so to get the distance in meters, we must use the arc formula (see Figure 9).

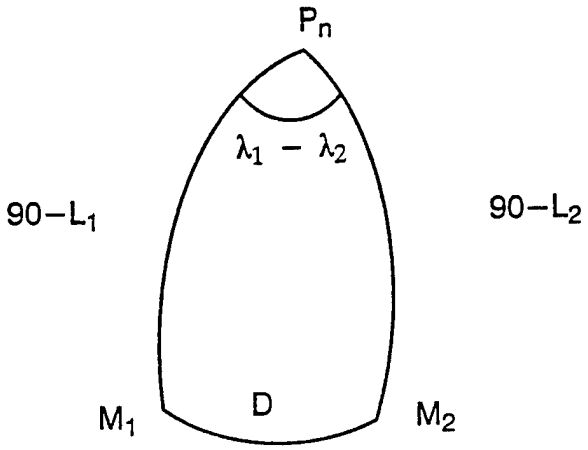

Figure 9. Spherical triangle.

$$s = .5 * (7915.6 * 1609.3) * D * (\pi/180)$$

* See Appendix A for more on UTM coordinates.

Note that here we are using 7915.6 miles as the Earth's diameter averaging the differing polar and equatorial diameters.

This calculation is lengthy and time consuming even for a computer, so it was decided that most positional operations would be performed in UTM coordinates.

For convenience, when a latitude and longitude position is received by the Distributed FactBase (DFB), it is translated into a set of UTM coordinates with a zone, easting, and northing.* Latitude, longitude, and UTM coordinates are then stored in the local database. However, when positional data must be sent between databases, either set of coordinates could be sent and the other recalculated at the destination.

3.2 Precision. GPS data is commonly considered to be accurate to 15 m (for military purposes) if sufficient satellites are visible. Differential GPS techniques may improve this to 10 m or even better depending on the quality of equipment being used and signal strengths. Since one of the design goals of the IDT data abstractions is to be able to service any military need, the precision we finally select must be adequate for all military situations. Since the Military Grid Reference System (MGRS), at its most precise, records position data to 1 m, let us determine the requirements to store positional data accurate to 1 m. Figure 10 shows the basic geometry.

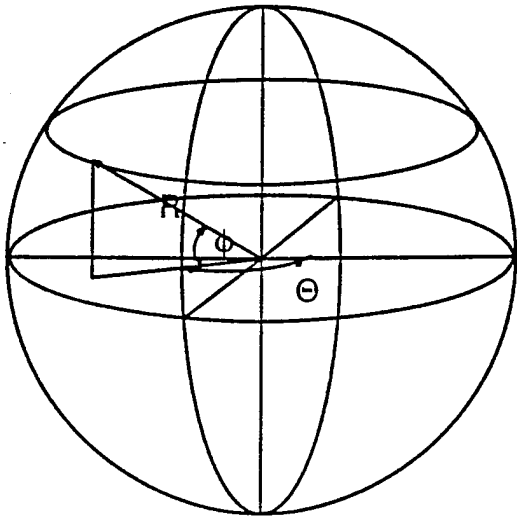

Figure 10. World geometry.

* This translation is accomplished by using functions obtained from the Sensors, Signatures, and Signals Directorate of ARL. This program is an adaptation of USGS Program J380, dated 8/14/1984.

The Earth has a diameter at the equator of approximately 7,926 miles. This is 12,755.8 km. Therefore, the circumference of the Earth at its equator is 40,073.577 km. To get 1-m accuracy requires breaking a circle into 40,073,577 parts. Since

25 bits = 2^25 = 33,554,432 "different pieces" and

26 bits = 2^26 = 67,108,864 "different pieces,"

26 bits is the smallest number of bits in a binary system that can individually represent 40,073,577 different pieces. So 26 bits covers the east–west dimension. Of course, this is the worst case since the distance around the world at a given latitude decreases as one moves from the equator to the poles.

The north–south direction requires one less bit because one only has to cover half a circle (from the North Pole to the South Pole). So 1-m accuracy requires approximately 20,036,788 pieces.

24 bits = 2^24 = 16,777,216 "different pieces" and

25 bits = 2^25 = 33,554,432 "different pieces."

So 25 bits is the smallest number of bits in a binary system that can individually represent 20,036,788 different pieces. In other words, 24 bits gives 1.19-m accuracy and 25 bits provides 60-cm accuracy.

Assuming R (range) needs to cover from the bottom of the oceans (~–7 miles) to geo-synchronous orbits (~22,400 miles), to cover 36,061,010 m requires at least 26 bits. So 25 bits provides 1.07-m accuracy, and 26 bits provides 53-cm accuracy. So 26 bits covers R. Note that the 1-m positional accuracy is at the Earth's surface. At the 22,400-mile orbit, our accuracy in the θ and ϕ directions is 3.8 m.

To review—1-m accuracy requires: 26 bits for longitude, 25 bits for latitude, 26 bits for "altitude," and 32 bits for 1 s time. Rounded to the nearest octet (8 bits, a standard), this is a total of 16 octets that offers more than enough resolution. As far as units, radians are a convenient unit for angles in spherical coordinates, meters are fine for distance (range), and seconds appears fine for time.

Since an integer is 32 bits on most modern machines, 3 integers will supply us with more than enough accuracy.

3.3 Fact Type Elements. The GPS unit we are using, the Garmin GPS 50 (Garmin International, Inc. 1992), returns latitude and longitude in degrees and minutes, with the minutes accurate to the nearest one-hundredth of a minute.

The location fact is currently defined:

```
define location {
    int     lat_min000      latitude in 1,000 of a minute
    int     lng_min000      longitude in 1,000 of a minute
    int     utm_zone        UTM zone number for this location
    int     utm_easting     UTM easting (meters)
    int     utm_northing    UTM northing (meters)
    int     altitude        altitude for current location (meters)
    int     gps_date        date of the year obtained from GPS unit
    int     gps_time        time of day obtained from GPS unit
    int     gps_status      status obtained from GPS unit
};
```

4. RAW DATA COLLECTION/GENERATION

4.1 Empirical Data Collection. Location data was collected from a Garmin 50 GPS (Garmin International, Inc. 1992) unit that was connected via 4,800 baud line to the GPS_RCV program running on a laptop computer. (See Appendix B for more information on the GPS_RCV program.) The data collected conforms to the NMEA 0183 format.* This is an ASCII format with predefined messages. For the purposes of this program, the message of interest is the RMC (recommended specific GPS data). The RMC message is received by the computer, along with a number of other messages that include such information as cross track error and autopilot information. These other NMEA sentences include GPBWC, GPGLL, GPRMB, GPRMC, GPR00, GPWPL, GPXTE, and the GARMIN proprietary sentence PGRMA.

The RMC formation is:

RMC,utc,A,latitude,N,longitude,W,xxx,xxx,data,xxx,e

* Version 1.5; December 1987.

Field	Description
1	Universal Time Coordinated
2	Status; A = valid, V = invalid
3,4	Latitude; N = north, S = south
5,6	Longitude, W = west, E = east
7	Speed over ground in knots
8	True course over ground
9	Date
10,11	Magnetic Variation; E = east, W = west

A couple of these fields require a little explanation. The Universal Time Coordinated field is the number of hours, minutes, and seconds from midnight in Greenwich Mean Time (GMT). For instance, 154239 is 15 hr 42 min and 39 s GMT. Similarly, the Date field is the date, month, and year (191193 = 19th of November 1993). Both the latitude and longitude fields have degrees and minutes encoded as a single floating point number with the decimal part representing the fractional part of minutes (generally hundredths: 3928.60 = 39°, 28.60 min).

One also notices that altitude is not available in the GPRMC message. To obtain the altitude value from the GARMIN GPS unit, the proprietary message PGRMZ must be decoded.

The PGRMZ message is:

$PGRMZ,altitude,f,p*cs

Field	Description
1	Present Altitude
2	Units (feet)
3	Position fix dimensions (2 = user altitude, 3 = GPS altitude)
4	Checksum.

If the number of available satellites falls below three, the latitude and longitude field are left blank and the status flag indicates an invalid entry.

A typical message set is:

```
$GPRMC,145141,A,3928.56,N,07605.12,W,000.5,339.3,070694,011.2,W*76
$GPRMB,A,,,,,,,,,,,V*71
$GPR00,,,,,,,,,,,,*45
$GPGLL,3928.56,N,07605.13,W*7C
$PGRMZ,278,f,3*16
$PGRMM,WGS 84  *26
$GPXTE,A,A,,,N*3C
$GPBWC,145142,,,,,,,T,,M,,N,*11
$GPVTG,339.3,T,350.4,M,000.5,N,001.0,K*42
$PSLIB,,,K*23
$PSLIB,,,J*22
```

The initial attempt to parse the RMC message made a number of simplifying assumptions. The major problem arises from the manner in which the serial interface with the computer delivers incoming information to the program. The bytes given to the program at any given time will be an arbitrary grouping of the above. So the byte set of buffer may begin with a message or anywhere within a message. The initial parser was very simplistic in that if a particular buffer didn't have a large number (400+) of bytes in it, it was thrown away in hopes that a larger grouping would soon be coming. This was done in order to make sure that the buffer always contained a GPRMC message. In addition, only one RMC message out of a buffer was located and processed. The number of messages thrown away by this method was excessive and actual practice resulted in significant time periods expiring without any updates being received by the DFB. Resolution of this problem was obtained by implementing a smarter parser that keeps track of where it is within a message.

4.2 Terrain Analysis Programs. While Figure 1 shows an axis of advance (AA), it is important to mention that leaders should make every attempt to give subordinates AAs that "make sense" so that deviations from the recommended AA are not because of trivial oversights. If leaders base the AA on a good terrain evaluation, the current enemy situation, and the size, type, formation, etc. of our own forces, we may prevent some deviation from the plan. Detailed terrain information immediately available to commanders, planners, or our C2 system is not just a future capability. The Program Evaluation Office (PEO) for Command and Control Systems and the Project Manager for Common Software currently sponsor a program called the Terrain Evaluation Module (TEM). TEM is an example of software that will enable our command and control system to provide the detailed terrain data required on the battlefield for computationally intensive situational awareness. TEM gets its data from digital map/terrain products

obtained from the Defense Mapping Agency (DMA). These computerized maps contain much more detailed surface feature information than paper maps. They contain information about obstacles, transportation systems, elevation, slope, vegetation, soil content, and surface drainage. Some of the capabilities include:

Display over the ground distance

Predict travel time in accordance with vehicle type and restrictions

Predict travel time according to unit composition, restrictions, etc.

Display mobility corridors and AA

Display air mobility corridors and AA in specified area

Display information about a specific location

Create overlays showing selected terrain characteristics

Display information concerning lines of communication (LOC)

Determine visual intervisibility between points using elevation data

Communications line of sight (LOS)

Determine/display areas within range for selected weapons types.

Many of these tasks are essential for the predictive solution to situational awareness that we are discussing. These capabilities do exist now and will become part of the Army Tactical Command and Control System (ATCCS). Moving this type of information from the planning to the execution phase of missions will allow us to take full advantage of these capabilities. The types of tactical decision aids (TDAs) possible with this type of system and the speed with which they may be produced will lend an incredible advantage to our forces in planning and execution.

4.3 Route Simplification. With the IDT predictive solution for situational awareness comes the requirement for each unit to share proposed route information with other units so that each unit can predict where the other units will be. This information must be brief enough to be distributed across the battlefield, but also detailed enough for accurate predictions to be made. In this section, an algorithm will be presented that generates "sufficiently accurate" route segments from more detailed data gathered from

reconnaissance, or route planning software. "Sufficiently accurate" will be defined by commander's guidance, the terrain, and the tactical situation.

As input data, the segment generator expects a set of entries each of which consists of X and Y UTM coordinates. In addition, the velocity and time are also required at each coordinate. Figure 11 shows a route created by putting a GPS unit, along with a computer to record the data, into a vehicle and collecting data for Spesutie Island on Aberdeen Proving Ground, MD. This is much like the route data a reconnaissance unit with GPS receivers might be expected to generate. The route begins in the bottom right-hand corner and progresses counterclockwise to the bottom left-hand corner and then returns up to the left-hand top. The overlapping route along the left side was actually produced by traveling along the same road. The differences are due to the inaccuracies in the GPS, such as selective availability. Given this information, the algorithm first computes segments of constant velocity. (Having such segments simplifies the prediction algorithm to a simple interpolation program.)

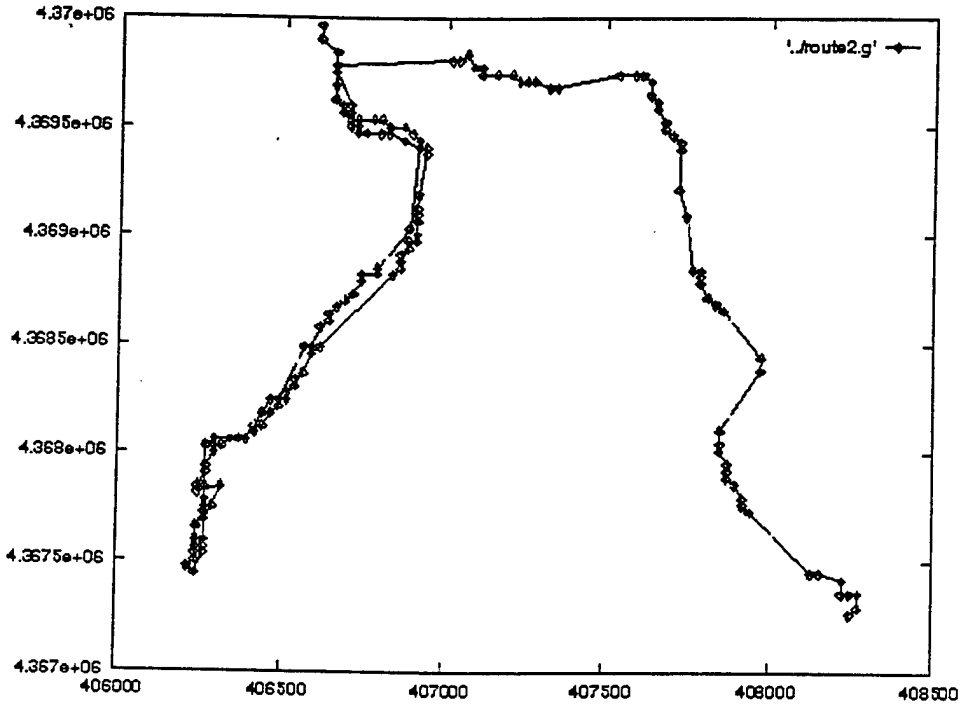

Figure 11. Spesutie Island empirical route data.

Figure 12 shows the velocity profile for this route. To create the constant velocity segments, a running average is computed and each new velocity is compared to that average. Each velocity is factored into the average before it is compared. If the velocity differs from the average by more than the specified tolerance, a new segment is begun. Figure 13 shows the segments generated for the velocity data in

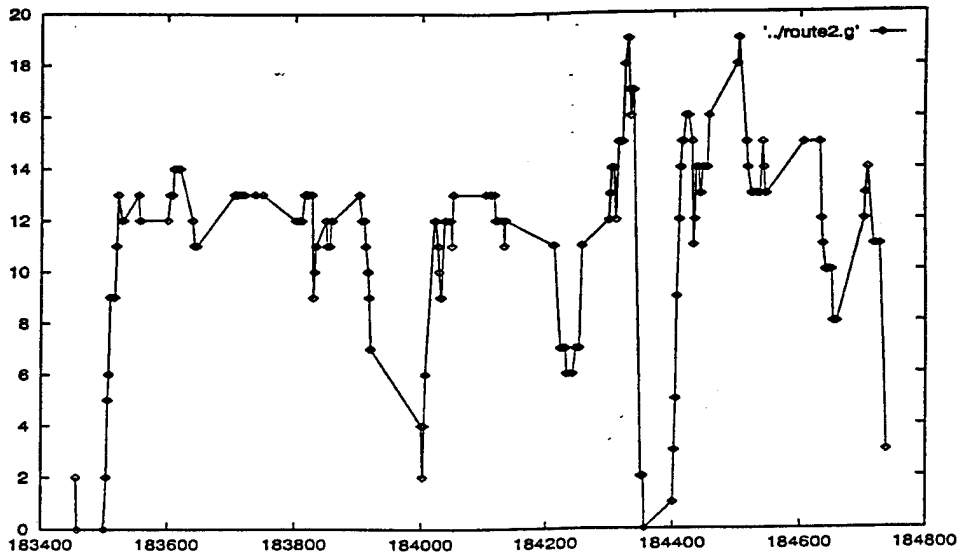

Figure 12. Spesutie Island empirical velocity data (m/s).

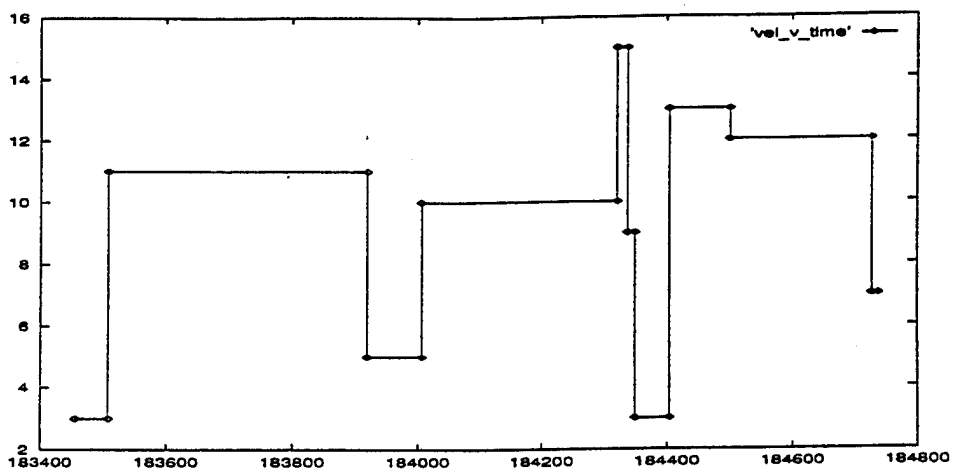

Figure 13. Constant velocity segments by the running average method.

Figure 12 using a tolerance factor of 5 m/s. An overlay of these segments on the original route is shown in Figure 14.

These segments are then used as input to the routine that further restricts the segments to conform to the specified positional tolerance. The algorithm used for this is again very straightforward. The first segment from the constant velocity procedure is taken, and the distance from each data point to this segment is computed. To perform this feat, we use the endpoints of the velocity segment as the two points in the two-point equation of a line.

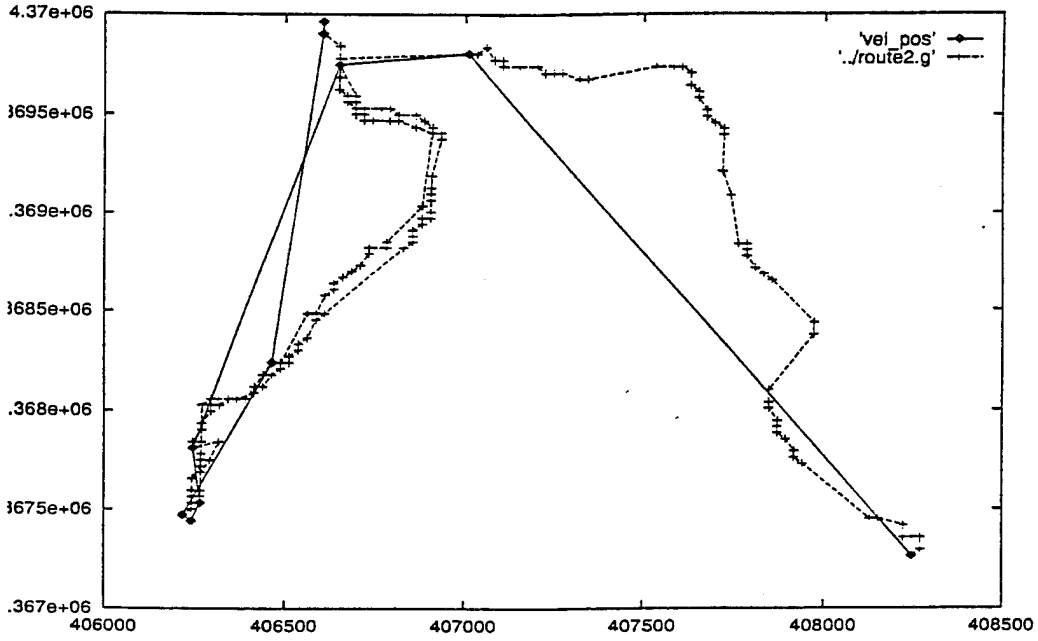

Figure 14. Spesutie Island route segments determined by constant velocity.

$$\frac{(x-x_1)}{(x_2-x_1)} = \frac{(y-y_1)}{(y_2-y_1)}$$

Solving for y and rearranging gives the appropriate coefficients for the distance from a point to a line equation.

$$y = \frac{(y_2-y_1)}{(x_2-x_1)}\,x - \frac{(y_2-y_1)}{(x_2-x_1)}\,x_1 + y_1$$

Now since the general equation for the distance from a point to a line is

$$d = \frac{|Ax_p + By_p + C|}{\sqrt{(A^2 + B^2)}},$$

we have

$$A = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

$$B = -1$$

$$C = y_1 - \frac{(y_2 - y_1)}{(x_2 - x_1)} x_1.$$

Using these values, we check each data point to see if it is within the positional tolerance specified. When a point is found to lie outside of this tolerance, a new segment is created using the out of bounds point as the endpoint of the original segment and the start point of the new segment. The endpoint of the original segment forms the endpoint of this new segment. Figure 15 shows this procedure. With the problem subdivided, the algorithm is called recursively with the new segments. Both segments must be considered since points that originally were within tolerance of the first line segment might be out of tolerance of the new segment. The final results of applying these procedures are shown in Figure 16. Here we portray the original route with the route generated using a velocity tolerance of 5 m/s and a positional tolerance of 200 m.

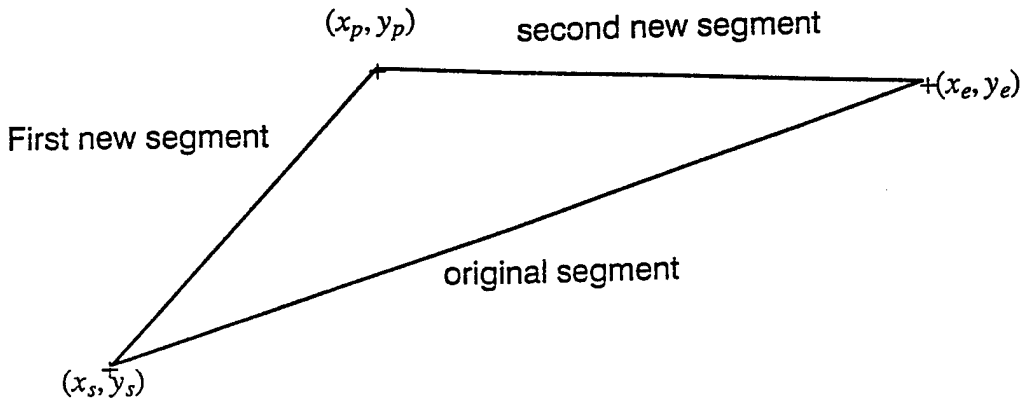

Figure 15. Segment subdivision.

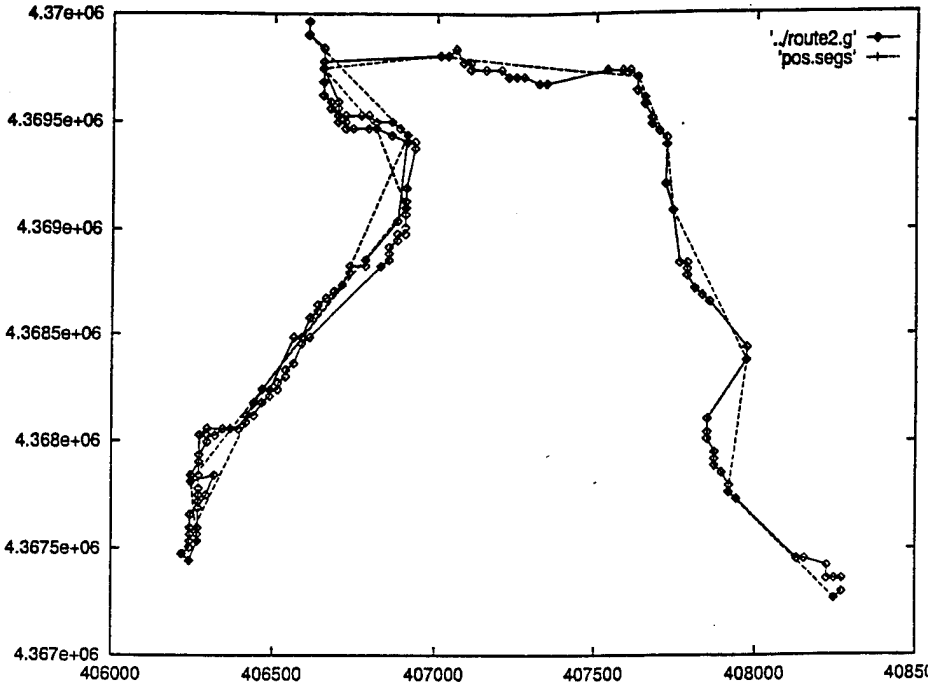

Figure 16. Positional constrained line segments.

Figure 17 presents the results of comparing the original route with positions predicted from the route segments generated in the previous section. In this figure, we have interpolated positions from the computed line segments and original route at constant time intervals and calculated the difference between this predicted location and the original route.

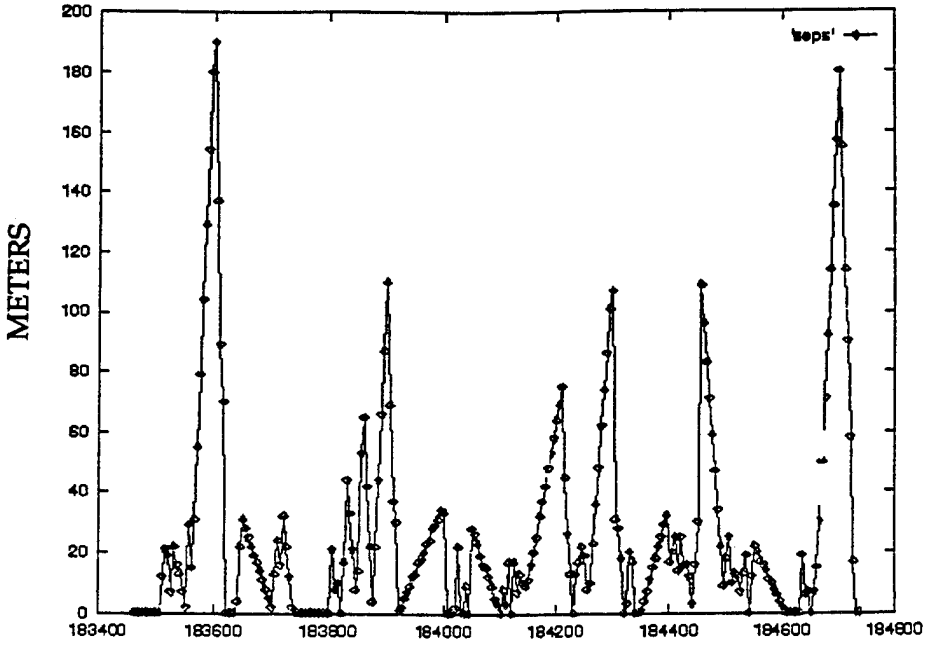

Figure 17. Differences between computed and original line segments.

4.4 User Inputs. An interactive program has been written to demonstrate how a route may be generated from raw GPS data. The program reads a file of position, time, and velocity values as generated by a GPS unit, then applies the smoothing algorithms as explained previously. A portion of the map window showing the data curves is in Figure 18. The darkest line shows the raw data, while the light line that crosses both Back Creek and Spesutie Narrows is the route based on constant velocity segments. The final, or positionally constrained, route connects the objectives, which are marked with open circles. (The portion of the route that doubles back on itself has been removed for clarity.)

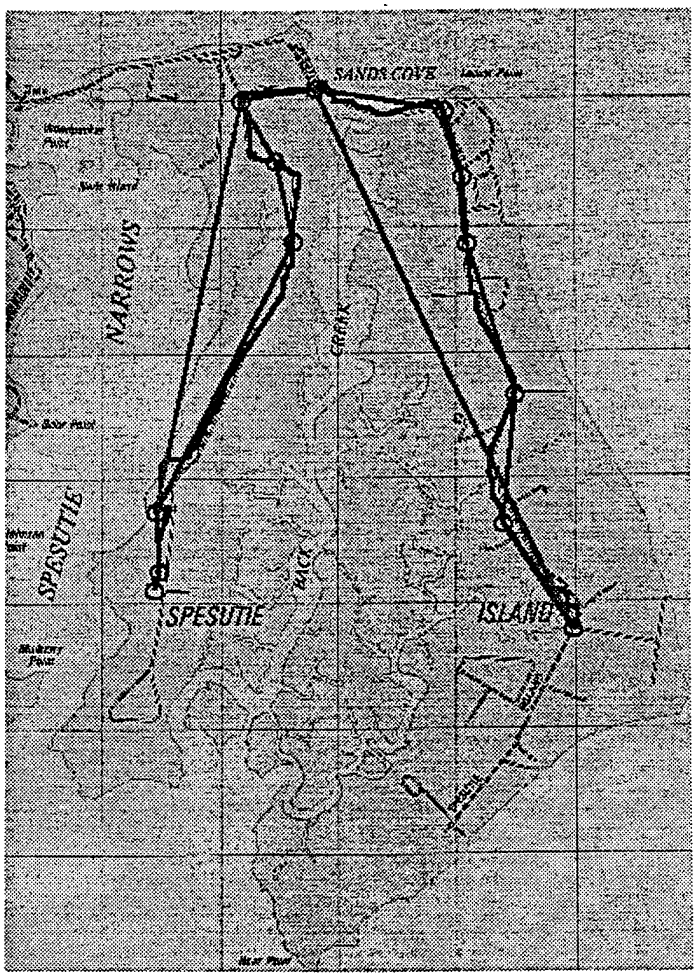

Figure 18. Spesutie Island showing raw data and routes.

The user may adjust the velocity and position tolerances and compute new routes based on them. For the sample data, there are 123 raw data points. Using a velocity tolerance of 5 m/s, 6 velocity segments were generated; a position tolerance of 200 m expanded that to 13 final segments. This reduces the number of points required to describe the route by a factor of 10, while sacrificing very little accuracy.

Decreasing the position tolerance to 100 m adds only 5 more segments for a total of 18, while tolerances of 241–345 m drop the route to 9 segments. While these numbers will vary with different sets of data, they give the user a feel for how free or restrictive an SOC should be.

Once the user has generated a route by applying the smoothing algorithms to the raw data, he may add more objectives. He may want to force the route to cross a bridge or restrict the route in some manner. He also defines all the parameters for the SOC for each segment. The left and right distances may be close to the position tolerance, while the early and late times will depend on the current plan. This final sequence of objectives, and their SOCs, are stored in the fact base and distributed to other units.

5. TACTICAL SIGNIFICANCE

5.1 Benefits. With the addition of operation order (OPORD) builders, terrain evaluation programs, computers, GPS devices, and tactical displays in vehicles or on individuals, our forces will have the ability to avoid the problems described by MG Scales on page 1. Recent advances in computational power and certain future advances will enable our forces to start each battle with a detailed, distributed, digital plan. But this plan will be much more powerful in its usefulness. First, we can do the planning much quicker and stay inside the decision-making cycle of the enemy; and second, we can distribute these plans quickly enabling our forces to start operations sooner. Commanders may also use the plan to follow the battle as it unfolds, resulting in other benefits, such as a dramatic reduction in the number of radio transmissions needed. With the prediction algorithms described earlier, our forces will know where other friendly forces are and can act swiftly and decisively while conducting planned missions or reacting to intelligence on a new enemy situation.

5.2 An Example. Rapid relay of battle drill instructions is one reward from this automated capability. Units use battle drills to provide quick responses to enemy actions. While these are immediate responses, there still needs to be some guidance from leaders while developing the situation and directing forces into the proper positions. These battle drills would be preloaded into the system so a leader would only have to initiate the battle drill with minimal input and instructions would appear on his subordinates' tactical displays. Much of the information needed will already be stored on the subordinates' computer: so, again, less communication is needed.

Take the example of a Mechanized Infantry Platoon moving along an AA. If this platoon makes contact, it has several options: it can bypass the enemy with permission from the commander, conduct a hasty attack, fix the enemy so another unit can conduct the assault, conduct a hasty defense, or establish

a hasty ambush. The general guidelines for these actions (whether doctrine, straight from a field manual, or from the platoon's standing operating procedures) can be "digitized" and already stored in the platoon's database. Some of the determinants of action include the size of the enemy element, the level of resistance they are providing, and how many friendly casualties they are inflicting. For this example, the enemy resistance is light and the decision is made to conduct a hasty attack.

The platoon leader initiates the battle drill for his unit to conduct a hasty attack on his computer and instantly the hasty attack graphics appear on displays across the platoon and company. (See Figure 19.) How he enters the information is another topic and will not be covered in this report. However, one can imagine many possibilities (e.g., voice recognition, touch screen). One section (Bradley fighting vehicles [BFVs] 1 and 2) knows that they are to provide supporting fires (by seeing their vehicle icons positioned in a support by fire position). BFVs 2 and 3 are to conduct the assault. This system could also be tied in with the terrain analysis software to suggest the best covered and concealed route to the objective. All of these actions can occur much faster than dismounting to give instructions or sending a radio message to get all these plans in place. This is just one example. It is easy to see that for different battle drills or tactical maneuvers, similar information can be devised. The displays for the remainder of the company could show these graphic symbols, a subset of them, or just the SOCs for the platoon vehicles. This may be a selectable option depending on what the commander would like to see. More research must be conducted to optimize exactly how and what makes sense to display.

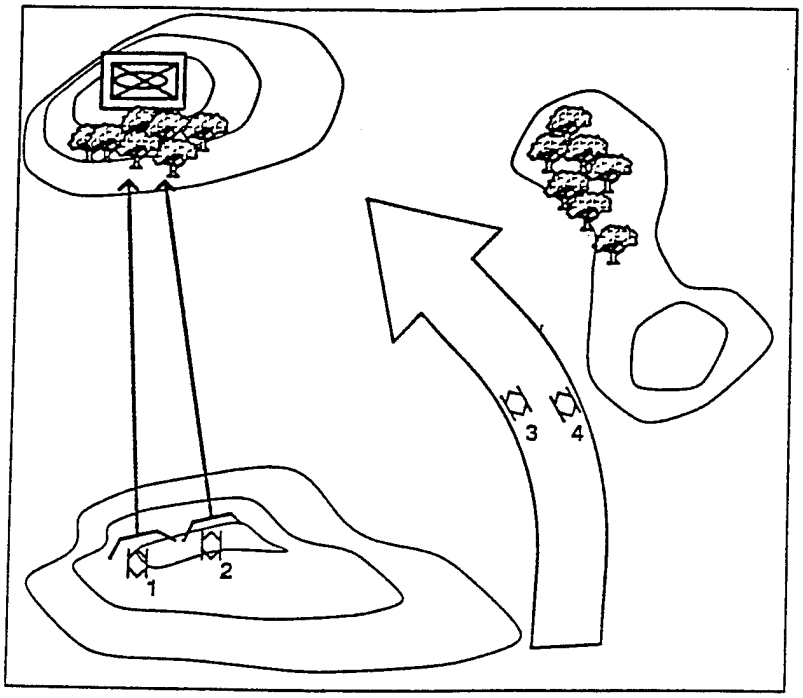

Figure 19. Hasty attack battle drill graphics.

6. CONCLUSIONS/FUTURE WORK

"We must be innovative and mentally agile. We must be critical thinkers and have the courage to break with tradition."

—MG John A. Dubia
Commandant
U.S. Army Field Artillery School
Field Artillery, August 1994.

6.1 Conclusions. GPS devices, which allow a unit to know precisely where it is, will have a major impact on the digital battlefield. However, they provide only positional awareness and not situational awareness. Radio bandwidth is severely limited and it is not practical for units to frequently report their positions. In addition, when communications fail or a GPS device cannot get an adequate fix to determine its location, other methods must be used to identify units.

The use of route planning to define a dynamic AA is a natural extension of current operating procedures. The Army is developing software that performs terrain analysis to generate routes, automating a difficult and time-consuming process. This detailed route information, or data collected by a scout unit with a GPS device, may be simplified into a series of waypoints.

Putting an SOC around the unit's predicted location gives it real-world flexibility while also providing a measure of control. If units exchange their planned routes before a mission, everyone may predict each other's positions with enough accuracy to prevent fratricide. Even without radio communications, value is added because plans and objectives have been distributed, not just real-time locations. These route plans are in fielded databases, not a centralized database with limited access.

Planned dynamic AAs, along with prediction models using an SOC, will provide situational awareness while also decreasing the need for radio communications.

6.2 Future Work. The methodology that has been developed must be implemented in a fieldable system for testing. The initial testbed will be the CAC2 ATD. This exercise will show both the practicality of SOCs and preplanned routes and the underlying communications and database techniques of the IDT.

More research must be done to determine the best shape to use for the SOCs. While complex shapes may accurately model a unit's movement, the predictive algorithms should not be too computationally intensive because a unit could be calculating its location as often as every two seconds.

As MG Dubia said, commanders must be willing to try new ideas. At the same time, programs to achieve situational awareness cannot ignore the way the Army conducts its business and need to acknowledge limitations in radio bandwidth and other resources on the battlefield.

7. REFERENCES

Dubia, MG J. A. "Field Artillery." U.S. Army Field Artillery School, p. 1, August 1994.

Garmin International, Inc. "GPS 50 Personal Navigator Owner's Manual." 190–00015-01, Rev. B, p. C-2, Lenexa, KS, October 1992.

Garmin International, Inc. "GBR 21 Beacon Receiver Owner's Manual." 190–00069-00, Rev. A, p. 1, Lenexa, KS, November 1993.

Scales, MG J. "Certain Victory: The U.S. Army in the Gulf War." Office of the Chief of Staff, United States Army, Washington, DC, 1993.

INTENTIONALLY LEFT BLANK.

APPENDIX A:

UNIVERSAL TRANSVERSE MERCATOR (UTM) COORDINATE SYSTEM

INTENTIONALLY LEFT BLANK.

UNIVERSAL TRANSVERSE MERCATOR (UTM) COORDINATE SYSTEM

Universal Transverse Mercator (UTM) coordinates provide a system of specifying locations on the surface of the globe. A UTM coordinate is composed of a *zone number*, an optional *zone character*, *easting*, and a *northing*. Zone numbers represent 6° longitudinal strips extending from 80° south latitude and 84° north latitude. They are numbered starting at 0° longitude and proceeding westward. Zone characters represent 8° zones extending north and south of the equator. Each zone has a central meridian. Eastings are measured from the central meridian with a 500-km offset to ensure positive numbers. Northings are measured from the equator with a 10,000-km offset for locations south of the equator. There are special zones between 0° and 36° longitude above 72° latitude and a special zone 32 between 56° and 64° north latitude.

The Military Grid Reference System (MGRS) is an extension of the UTM system. UTM zone number and zone character are used to identify an area 6° in east–west extent and 8° in north–south extent. UTM zone number and designator are followed by 100-km-square easting and northing identifiers. The system uses a set of alphabetic characters for the 100-km grid squares. Starting at the 180° meridian, the characters A to Z (omitting I and O) are used for 18° before starting over. From the equator north, the characters A to V (omitting I and O) are used for 100-km squares, repeating every 2,000 km. Odd-numbered UTM zones begin at the equator. Even-numbered zones offset the rows by five characters. South of the equator, the characters continue the pattern set north of the equator. Complicating the system, ellipsoid junctions (spheroid junctions in the terminology of MGRS) require a shift of 10 characters in the northing 100-km grid square designators. Different geodetic datums using different reference ellipsoids use different starting row offset numbers to accomplish this. MGRS UTM zone number, UTM zone, and the two 100-km-square characters are followed by an even number of numeric characters representing easting and northing values. If 10 numeric characters are used, a precision of 1 m is assumed. Two characters imply a precision of 10 km. From 2 to 10 numerical characters, the precision changes from 10 km, 1 km, 100 m, 10 m, to 1 m.[1]

---

[1] Dana, P. H. World Wide Web page: **http://wwwhost.cc.utexas.edu/ftp/pub/grg/main.html**, Department of Geography, University of Texas at Austin, TX, 23 November 1994.

INTENTIONALLY LEFT BLANK.

APPENDIX B:

GPS_RCV MANUAL PAGE

INTENTIONALLY LEFT BLANK.

NAME GPS_RCV - GPS interface for the IDT

SYNOPSIS gps_rcv [-b lo] [-a] [-d]

OPTIONS

[-b lo] This flag causes the interface program to expect input on tty A at 4,800 baud. If this flag is absent, 9,600 baud is assumed.

[-d] Turns on debugging prints. The output can be very verbose so use with caution.

[-a] Connect to a Distributed FactBase on the local host.

GPS_RCV Currently uses no variables from the environment.

DESCRIPTION

Currently the GPS_RCV software executes on Sun Microsystems computers under the Solaris 1.03 Unix (Unix is a trademark of AT&T) Operating System.

THE INTERFACE

The Garmin 50 GPS unit sends data to the computer at 4,800 baud. This data is in accordance with the NMEA 0183 (Version 1.5; December 1987) format. This is an ASCII format with predefined messages. For the purposes of this program, the message of interest is the RMC (recommended specific GPS data). The RMC message is received by the computer along with a number of other messages that include such information as cross track error and autopilot information. These NMEA sentences are GPBWC, GPGLL, GPRMB, GPRMC, GPR00, GPWPL, GPXTE, and the proprietary sentence PGRMA.

The RMC formation is as follows:

RMC,utc,,latitude,N,longitude,W,xxx,xxx,data,xxx,E

Field	Description
1	UTM
2	Status; A = valid, V = invalid
3,4	Latitude; N = north, S = south
5,6	Longitude; W = west, E = east
7	Speed over ground in knots
8	True course over ground
9	Date
10,11	Magnetic Variation; E = east, W = west

A couple of these fields require a little explanation. The Universal Transverse Mercator (UTM) field is the number of hours, minutes, and seconds from midnight in Greenwich Mean Time (GMT). For instance, 154239 is 15 hr 42 min and 39 s GMT. Similarly, the Date field is the date, month, and year (191193 = 19th of November 1993). Both the latitude and longitude fields have degrees and minutes encoded as a single floating point number with the decimal part representing the fractional part of minutes (generally hundredths: 3928.60 = 39°, 28.60 min).

One also notices that altitude is not available in the GPRMC message. To obtain the altitude value from the GARMIN GPS unit, the proprietary message PGRMZ must be decoded.

The PGRMZ message is $PGRMZ,altitude,f,p*cs

Field	Description
1	Present altitude
2	Units (feet)
3	Position fix dimensions (2 = user altitude, 3 = GPS altitude)
4	Checksum

If the number of available satellites falls below three, the latitude and longitude fields are left blank and the status flag indicates an invalid entry.

A typical message set is shown:

```
$GPRMC,145141,A,3928.56,N,07605.13,W,000.5,339.3,070694,011.2,W*76
$GPRMB,A,,,,,,,,,,,V*71  $GPR00,,,,,,,,,,,,*45
$GPGLL,3928.56,N,07605.13,W*7C  $PGRMZ,278,f,3*16
$PGRMM,WGS  84  *26  $GPXTE,A,A,,,N*3C
$GPBWC,145142,,,,,,T,,M,,N*11
$GPVTG,339.3,T,350.4,M,000.5,N,001.0,K*42
$PSLIB,,,K*23  $PSLIB,,,J*22
```

FILES

none

SEE ALSO

dfb(1)

BUGS

Note the GPS_RCV is research software and as such contains many bugs and inefficiencies.

APPENDIX C:

DFB MANUAL PAGE

INTENTIONALLY LEFT BLANK.

NAME dfb - Distributed FactBase (DFB) - an Information Distribution System (IDS) node

SYNOPSIS dfb [-i] [-n] [-c filename] [-p# filename] [-f]

OPTIONS

-i the presence of this flag causes the DFB to accept commands from the controlling tty.

-n causes the DFB to continue running even if all clients hang up. Normally the DFB will terminate when the last client closes its connection.

-c filename "-c" allows the user to specify a capabilities (or configuration) file for use during this execution of the DFB. If this flag is not present, the default file "cap_input_file" will be used. In the case where no capabilities file is present, the DFB will terminate.

-p# filename turns on package logging. If "#" = 1, logging is done in a terse form. If "#" > 1, then logging is considerably more verbose. "filename" indicates the file where logging is written.

-f turns off Fact Exchange Protocol (FEP) logging.

ENVIRONMENT

The DFB currently uses no variables from the environment.

DESCRIPTION

The DFB is a node in the IDS. It stores data, allows user manipulation of that data, and controls distribution of the data to other nodes in the system. Each DFB is composed of a freeform RAM-resident FactBase to store information, a Security Control Module (SCM) to control the flow of information in and out of the FactBase, an FEP that efficiently handles inter-DFB fact exchanges, and an interface that allows sophisticated application programs to communicate with the DFB. Currently, the DFB software executes on Sun Microsystems computers under the Sun 3.4 Unix (Unix is a trademark of AT&T) Operating System.

The FactBase management module is responsible for efficiently handling the storage and retrieval of information. To expedite the performance of these endeavors, the entire FactBase is maintained in memory while the DFB is operating. A small but sufficient query language is supported as well as an automatic notification scheme known as triggers. Triggers are a method by which interacting programs

may request a notification be sent them when particular data is altered. Messages received by the DFB are interpreted by a YACC (see YACC(1)) based parser.

Capability Profile File

The configuration of each DFB is controlled by a Capability Profile (CAP) file. This file defines the environment in which the DFB is to interact. It also defines the extent to which these interactions may occur. Contained within the CAP file are such diverse items as other nodes with whom interactions are allowed, specifications of communication channels, and rules which state actions to be performed when certain messages arrive.

The CAP file is organized in several sections as delineated:

I. Known Units
II. Channel Media
III. Connection Commands
IV. Broadcast Initiation
V. Overhearing Specifications
VI. Data Load Commands
VII. Identification Commands
VIII. Distribution Rules

The individual lines in the CAP input file are of the general form

keyword data

where keyword is one of

UNIT	Describes a unit that is included in this DFB's environment. This doesn't imply that the DFB will be allowed to communicate with it.
CONNECT	Specifies those units with which this DFB can currently communicate.
CHANNEL	Describes a communication data link that connects this DFB with one or more other DFBs.

OVERHEAR	Indicates from which unit the DFB will accept overheard information. This capability is implemented to allow the DFB to ignore messages sent from a scenario driver to another unit.
UDP_BROADCAST	Instructs the FEP to use the IP broadcast address, xxx.xxx.xxx.255, when transmitting on the ethernet.
LOAD	Causes the DFB to load a specified file as data. Note that this command is actually a member of the query language.
RULE	Defines the actions to be taken by the DFB when specified criteria are met. These actions may include the construction of trigger commands.
IAM	Defines who this node represents.
SUBUNIT	Defines the subunits of this node.
ADJACENT	Defines this node's immediate superior.
PARENT	Allows the user to place comments in the CAP input file.

Since these commands are identical in format to the query language, the syntax of the CAP commands is described along with the query language. Note that the order of the commands within the file is important since some of the CAP commands require information from previous lines.

In the following paragraphs, the query language syntax as well as the syntax for the CAP's commands will be presented.

In the UNIT declaration section, all units this DFB should or may need to know about are declared. The table built from these lines is used to look up addresses given unit names. So any unit not defined in this section will be unable to communicate with this DFB. Syntax is

UNIT unit_name host_name host_addr tries time_out window_size,

where

unit_name	The name of the unit.
host_name	The name of the host computer for this unit.

host_addr	The internet address for this host.
tries	The number of times the FEP will try to send this message if an acknowledgment from the receiving host is not received. This number should be at least one.
time_out	The time in seconds. This number plus the channel timeout equals the time to wait for an acknowledgment from the receiving node.
window_size	A variable used by the FEP. If the number of messages sent to this unit that have not been acknowledged exceeds this number, do not send any more messages to this unit at this time. Wait and try again.

Examples of typical unit declarations are

UNIT "2ND BDE FSE" "sabre.brl.mil" "192.5.23.120" 3 20 10

UNIT "2–51 FA BN TOC (OPNS)" "sable.brl.mil" "192.5.23.121" 1 8 10

UNIT "2–11 BN FSE" "satin.brl.mil" "192.5.23.130" 3 8 10

After all units are declared, it is necessary to describe the communication media available for inter-DFB communications. This is one via CHANNEL commands. Syntax for this type of command is

CHANNEL cname type tries timeout mtu window_size speed port

where

cname	A string name of the channel.
type	An integer name identifying the channel.
tries	The number of times the FEP will try to send this message if any acknowledgment from some receiving host on this channel is not received. This number should be at least one.
timeout	Time in seconds; this number plus the host timeout equals the time to wait for an acknowledgment.
mtu	Maximum transmission unit - the maximum size of the package to be sent on this channel.

window_size	A variable used by the FEP. If the number of messages sent to this channel that have not been acknowledged exceeds this number, do not send any more messages on this channel this time. Wait and try again.
speed	The baud rate of this channel.
port	The serial port where this channel is connected.

Typical channel declarations are

CHANNEL "UDP–1" 0 3 1 4096 8 10000000 "NONE"

CHANNEL "BB–1" 1 3 30 2048 8 1200 "/dev/ttya"

CHANNEL "SRL–1" 2 3 5 4096 8 9600 "/dev/ttyb"

After units and communication channels have been defined, CONNECT lines are used to specify those units with which this DFB will currently communicate and the channel by which they may be reached. The syntax is

CONNECT unit_name emcon_mode channel_name

where

unit_name	The name of the unit.
emcon_mode	In emission control mode? 1 = YES, 0 = NO.
channel_name	The string name of the channel on which the specified unit can be reached.

An example of a typical CONNECT declaration

CONNECT "2–51 FA BN TOC (OPNS)" 0 "UDP–1"

Next, the overhear command is defined. The messages from a unit not defined in the OVERHEAR command are discarded. At first glance, to selectively decide from whom the DFB should overhear messages from seems to contradict the IDS concept. This capability is provided for use in conjunction

with a scenario driver, where it is necessary to ignore overheard information sent from the scenario driver to a specific node. Otherwise, all nodes will have concurrent databases and this would skew the experiment. The syntax is

OVERHEAR unit_name

At this point, it becomes necessary to load at least the unit data since CAP inputs beyond this point often require organizational data contained in unit facts. The command used is

load "data_file"

The identification portion of the CAP file is where this DFB is told who it is. This information should match what appeared in the Units section of the CAP file.

IAM actual_unit reporting_unit rollup_factor

where

actual_unit	Fictitious unit this DFB is representing (e.g., brigade).
reporting_unit	Real (physical) unit this DFB is representing (e.g., brigade commander).
rollup_factor	Level of echelons below the host unit of a DFB at which the units are reporting to this host.

In addition, the reporting environment is described. The commands used to accomplish this are

SUBUNIT "unit_name;unit_name;unit_name;. . . "

ADJACENT "unit_name;unit_name;unit_name;. . ."

PARENT "unit_name"

where

unit_name	Name of the unit. These unit name(s) are separated by a semicolon.

Some examples are

IAM "2ND BDE FSE" "2ND BDE (2X2)/32D AD" 2

SUBUNIT "2–51 FA BN TOC (OPNS); 2–11 BN FSE; 2–92 BN FSE; 2–94 BN FSE"

Now the RULE section of the CAP file will be described. Rules are used to specify how information flows out the DFB. In general, each rule has the following structure

RULE ident ref_facttype {expr} {(action)[(action)} . . .]}

where

RULE	A keyword which must be present.
ident	An arbitrary string used to reference the particular rule in which it appears.
ref_facttype	A predefined facttype used to classify variables found in the incoming message.
ref_factid	If present, this rule will only be evaluated when this fact is altered. Note this feature is expected to be primarily for future use.
expr	This is an expression composed of variables and constants connected by arithmetic and Boolean operators. A number of functions are available.
action	Specifies the action that is to occur if this rule fires. For currently implemented distribution rules, the keywords SEND and TO denote what is to be sent and to whom it goes.

The following may appear as variables in the expression part of RULES

m_string	Data about message (i.e., source, destination, message type).
t_string	Variable names extracted from the incoming message text.
l_string	The last reported value of a data item.
fact_string	Special info about the pseudofact.
numbers	Constants.

Some simple examples of rules are

```
RULE "rule1" sensing { ($.m_type == 0x2000) }
  "{(SEND sensing, sensing.window,sensing.window.utm_loc
  TO\"2–51 FA BN TOC (OPNS)\")}"

RULE "rule2" line { ($.m_type == 0x4000) && ($.usage == "window") &&
  ((abs($.t_stime - $.1_stime) > 60) 11 (abs($.t_etime - $.1_etime) > 60)) }
  "{(SEND msg TO \"2–51 FA BN TOC (OPNS)\")}"
```

The Query Language

The commands understood by the DFB are noted:

dump
load filename
debug integer See following section.
erase
get fact_reference(s)
get_info fact_reference(s)
define fact_name {var_type var_name, . . . }
killfact fact_reference
killtrigger trigger_handle send trigger_handle dest, dest, . . .
indent LEX_STRING
update fact_reference [opt_fact_name] { opt_var_type var_name = var_value;
. . . }state fact_name { opt_var_type var_name = var_value; . . . }
var_type var_name = expr
test fact_reference {expr}
trigger trigger_handle fact_id . . . (expr)
let var_ref = expr
query fact_name . . . { expr }

Interactive Debugging

A number of levels of debugging prints are available to the interactive user. These are shown below and may be "or"ed together as desired. Note: It is very easy to cause thousands of lines of input to be generated. Perhaps the most useful of all the debugging options is DEBUG_PARSE; this results in the DFB printing out every command line it receives. It is invoked by the command

debug 0x40

Other debugging options are

DEBUG_TREE	0x01	Debug expression trees
DEBUG_LEX	0x02	Debug lexical analyzer
DEBUG_FACT	0x04	Debug fact/dd changes
DEBUG_TYPE	0x08	Debug expression types
DEBUG_MALLOC	0x10	Debug memory allocation
DEBUG_LIST	0x20	Debug lists
DEBUG_PARSE	0x40	Debug parsing
DEBUG_RULE	0x80	Debug rule processing
DEBUG_FEP	0x100	Debug FEP message processing
DEBUG_BB	0x200	Debug radio message processing

FILES

cap_input_file,fep_log_file,fep_stat_file

SEE ALSO

wmap(1), rg_chart(1), oplan(1)

BUGS

Note the DFB is research software and as such contains many bugs and inefficiencies.

INTENTIONALLY LEFT BLANK.

APPENDIX D:

SISOC MANUAL PAGE

INTENTIONALLY LEFT BLANK.

NAME sisoc - Spesutie Island Shape of Certainty (SOC) simulation

SYNOPSIS **sisoc** [**-gps** *gps_data_file*] [**-sim** *simulation_data_file*] [**-x** *hex_flags*]

AVAILABILITY

Currently, the prototype program runs on Sun Microsystems computers under the Solaris 2.x operating system with Motif 1.2.x.

DESCRIPTION

sisoc is a testbed program that operates in two modes. It reads in a file of raw Global Positioning System (GPS) data consisting of x,y coordinates, time, and velocity, then generates two routes. The first route is based on segments of constant velocity, within predefined bounds, and the second refines the route by constraining the position within another set of bounds. The raw data and both routes are drawn on a map of Spesutie Island, and the waypoints are marked with circles. The user may manipulate a pair of sliders to change the tolerances and generate new routes.

When the user has indicated the route is acceptable, **sisoc** enters its second mode. A simulation is run in real time, or a user-selected fraction or multiple of real time. The GPS data, or optionally a file of x,y,t data, is used to simulate the unit's position as a function of time. The actual location is marked with a solid circle, while a circular SOC is drawn at the predicted coordinates. A message is displayed indicating whether or not the unit is within the SOC; in addition, the SOC changes color and a bell may sound if it is outside the SOC. The user may stop the simulation, restart if from any simulation point, or cause it to run indefinitely.

OPTIONS

sisoc accepts all of the standard X11 options along with these additional options:

-gps *gps_data_file* Required argument at this time. Specifies the name of the file containing GPS data that will be smoothed into a route.

-**sim** *simulation_data_file* Specifies the name of the file containing data that simulates a unit moving along the route.

-**x** *hex_flags* hexadecimal value indicating which debug messages should be sent to **stderr**. The final value is obtained by "OR"ing the desired debug flags together.

0x0001	Print major X11 actions, such as creating widgets.
0x0002	Print actions involving colors and pixmaps.
0x0004	Print program arguments after processing.
0x0008	Print intermediate values computed by route smoothing algorithm.
0x0010	Print X11 expose events.

FILES

none

SEE ALSO

GPS_RCV(1)

NOTES

This is an early prototype. Future versions will connect to a Distributed FactBase and allow the user to interactively modify the generated route.

NO. OF COPIES	ORGANIZATION
2	DEFENSE TECHNICAL INFO CTR ATTN DTIC DDA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TA 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TL 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TP 2800 POWDER MILL RD ADELPHI MD 20783-1145
	ABERDEEN PROVING GROUND
2	DIR USARL ATTN AMSRL OP AP L (305)

NO. OF COPIES	ORGANIZATION
1	HQDA USAAIC ARMY 107 PENTAGON WASHINGTON DC 20310-0107
1	DISC4 ARMY 107 PENTAGON ATTN DAIS ADO WASHINGTON DC 2010-0107
1	CDR ARMY DIGITIZATION OFFICE ATTN HQDACS ADO 1745 JEFFERSON DAVIS HWY CRYSTAL SQUARE 4 STE 403 ARLINGTON VA 22202
1	CDR USAARMC ATTN ATSB-CG FORT KNOX KY 40121
1	CDR USAARMC ATTN ATZD CDS MAJ POWERS FORT KNOX KY 40121
1	CDR USAARMC ATTN ATZK MW FORT KNOX KY 40120
1	CDR USAIC ATTN ATSB CG FORT BENNING GA 31905
1	CDR TOPOGRAPHIC ENGRG CENTER ATTN CETEC TD AG 7701 TELEGRAPH ROAD ALEXANDRIA VA 22315-3864
1	CDR HQ TRADOC ATTN ATCS X BLDG 133 FORT MONROE VA 23651-5000
1	CDR HQ TRADOC ATTN ATCD ZA FORT MONROE VA 23651-5000

NO. OF COPIES	ORGANIZATION
1	CDR USAICS ATTN ATZS CD FORT HUACHUCA AZ 85613-6000
1	CDR USASIGCEN ATTN ATZH CDC FORT GORDON GA 30905
1	CDR USASIGCEN ATTN ATZHH BLT FORT GORDON GA 30905
1	CMDT USAFASCH ATTN ATSF CBL FORT SILL OK 73503-5600
1	PEO CCS ATTN SFAE CC SEO FORT MONMOUTH NJ 07703-5207
1	PM AFAS ATTN SFAE CC INT TSE 1616 ANDERSON RD MCLEAN VA 22102-1616
1	PM FATDS ATTN SAFAE CC FS TMD FORT MONMOUTH NJ 07703-5207
1	PM OPTADS ATTN SFAE CC MVR TM FORT MONMOUTH NJ 07703-5207
1	DIR ATTN AMSEL RD CS BC CC 2 SALTON CECOM FORT MONMOUTH NJ 07703-5207

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
21	DIR USARL ATTN AMSRL CI CA B D BROOME J C DUMER T P HANRATTY R A HELFMAN AMSRL CI CC A E M BRODEEN F S BRUNDICK H CATON S C CHAMBERLAIN A B COOPER III A R BOWNS D A GWYN G W HARTWIG JR R C KASTE M C LOPEZ M J MARKOWSKI C T RETTER L F WRENCHER S D KOTHENBEUTEL AMSRL CI CD J D GANTT AMSRL SS IC P EMMERMAN L TOKARSIK

INTENTIONALLY LEFT BLANK.

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author ARL-TR-1139 (Hartwig) Date of Report July 1996

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001,APG,MD

POSTAGE WILL BE PAID BY ADDRESSEE

DIRECTOR
U.S. ARMY RESEARCH LABORATORY
ATTN: AMSRL-IS-TP
ABERDEEN PROVING GROUND, MD 21005-5067